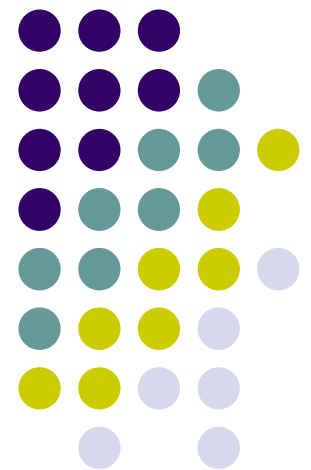
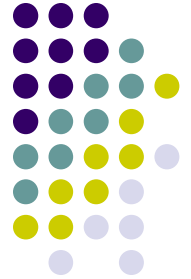


Universidad Nacional Autónoma de México
Facultad de Estudios Superiores Aragón

Ingeniería de Software

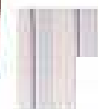
Clases y Objetos

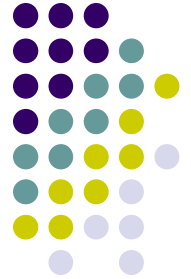



















Ejercicio

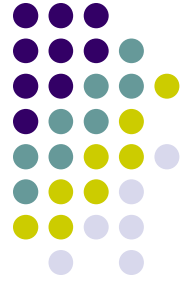
- Se te a encargado diseñar una clase para representar camisas para una aplicación de ventas en una tienda de ropa. Las camisas pueden ser de diferente tipo y diferentes precios.
- Se te proporcionan unas fotos para orientarte.
- El ejercicio consiste en diseñar la clase Camisa con notación UML.
- Tomar en cuenta que solo se requieren los atributos y métodos que sirvan para la aplicación de ventas





Resultado posible

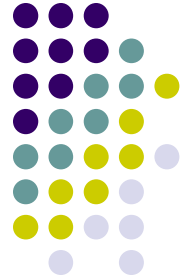
Camisa	
	color
	tipoMangas
	bolsas
	tipoTela
	precio
	set Color()
	set TipoMangas()
	set Bolsas()
	set TipoTela()
	set Precio()
	get Color()
	get TipoMangas()
	isMangas()
	get TipoTela()
	get Precio()



Ejercicio 2

Diseñar las clases relacionadas con el siguiente enunciado:

- Se requiere un sistema que me permita manejar el inventario de un almacén de muebles nuevos, los muebles pueden ser mesas, sillas, mesas de centro, sillones y roperos.
- Los muebles son de diferentes materiales y colores .
- En el caso de las sillas pueden ser con ruedas y la cantidad de ruedas puede variar. Hay mesas que son de 4 y otras de 6 patas.
- Se pretende que el sistema almacene la cantidad de cada uno de los tipos de muebles.
- El sistema permitirá registrar las salidas de los muebles de la bodega, especificando el tipo de mueble y la cantidad que salió.
- Cuando se saquen muebles de un tipo y la nueva cantidad disponible sea menor de 10, el sistema avisará para que se pidan mas.
- El sistema permitirá registrar nuevos muebles (que llegan de fabrica)
- El sistema realizará listas de las existencias en la bodega.



Alto nivel de abstracción

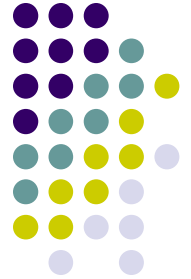
Almacen

ropero





silla


mesa









sillon











Bajo Nivel de abstracción

Almacen	
	name
	generarLista()
	sacarMuebles()
	meterMuebles()

Ropero	
	material
	color
	alto
	ancho
	setMaterial()
	getMAterial()
	opname()

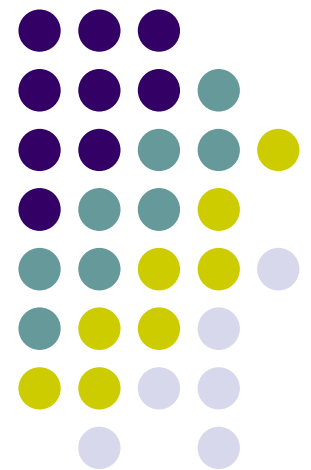
Silla	
	material
	color
	alto
	ancho
	tieneRuedas
	cantidadRuedas
	setMaterial()
	getMAterial()

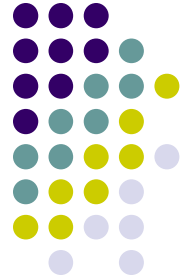
Mesa	
	material
	color
	alto
	ancho
	cantidadPatas
	esDeCentro
	setMaterial()
	getMaterial()

Sillon	
	material
	color
	alto
	ancho
	setMaterial()
	getMaterial()

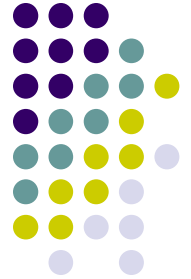
Diagramas de clases(UML)

Clases, niveles de abstracción y
relaciones





- El Diagrama de Clases es el principal producto del análisis y diseño del sistema
- La definición de clase incluye definiciones para atributos y operaciones
- El modelo de casos de uso debería aportar información para establecer las clases, objetos, atributos y operaciones
- presenta las clases del sistema con sus relaciones estructurales y de herencia



UML :Diagramas de clases

- Existen de dos tipos y cada uno con un objetivo claro:
- Diagrama con alto nivel de abstracción
- Diagrama con bajo nivel de abstracción

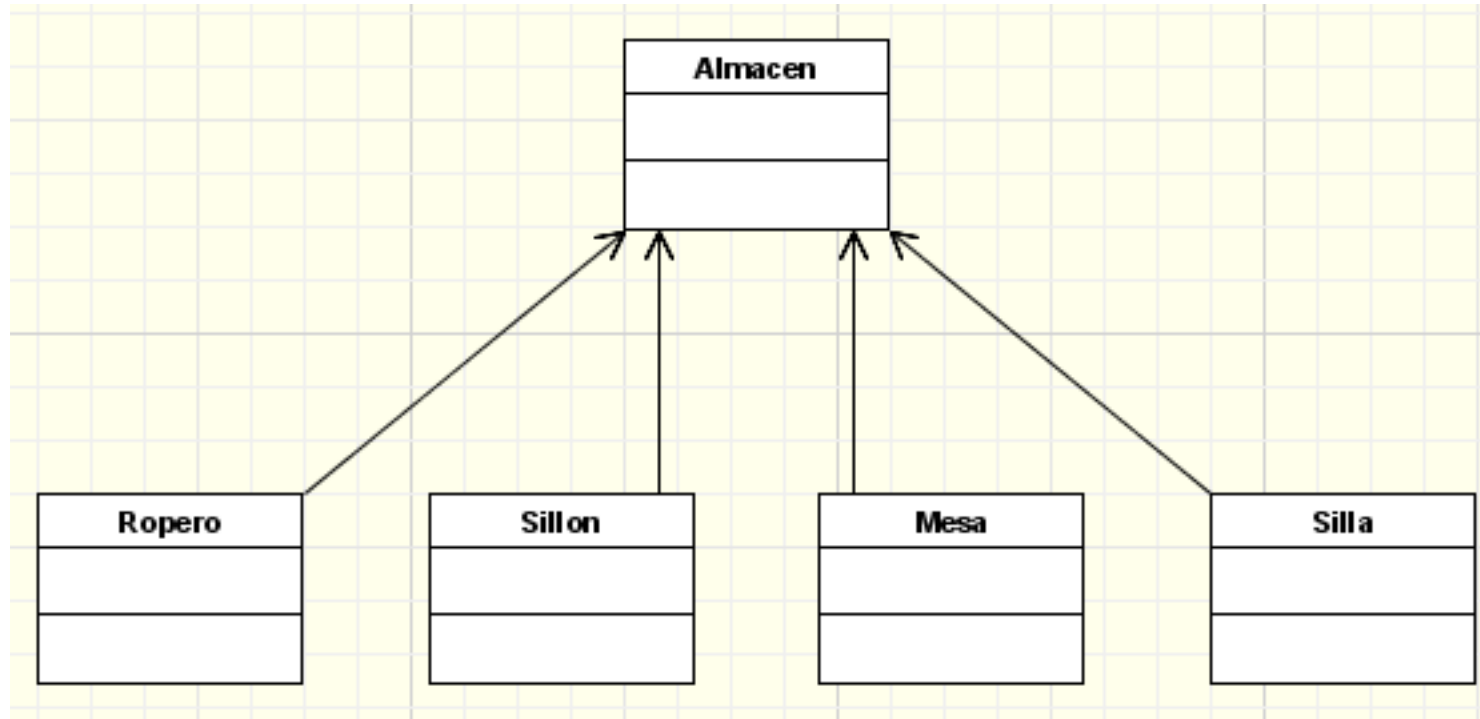


Diseño con alto nivel de abstracción

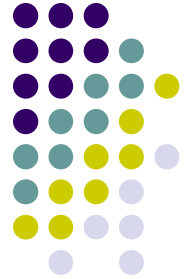


- Utilizada para determinar las clases involucradas en el sistema.
- También para establecer sus relaciones.
- En ocasiones su cardinalidad.

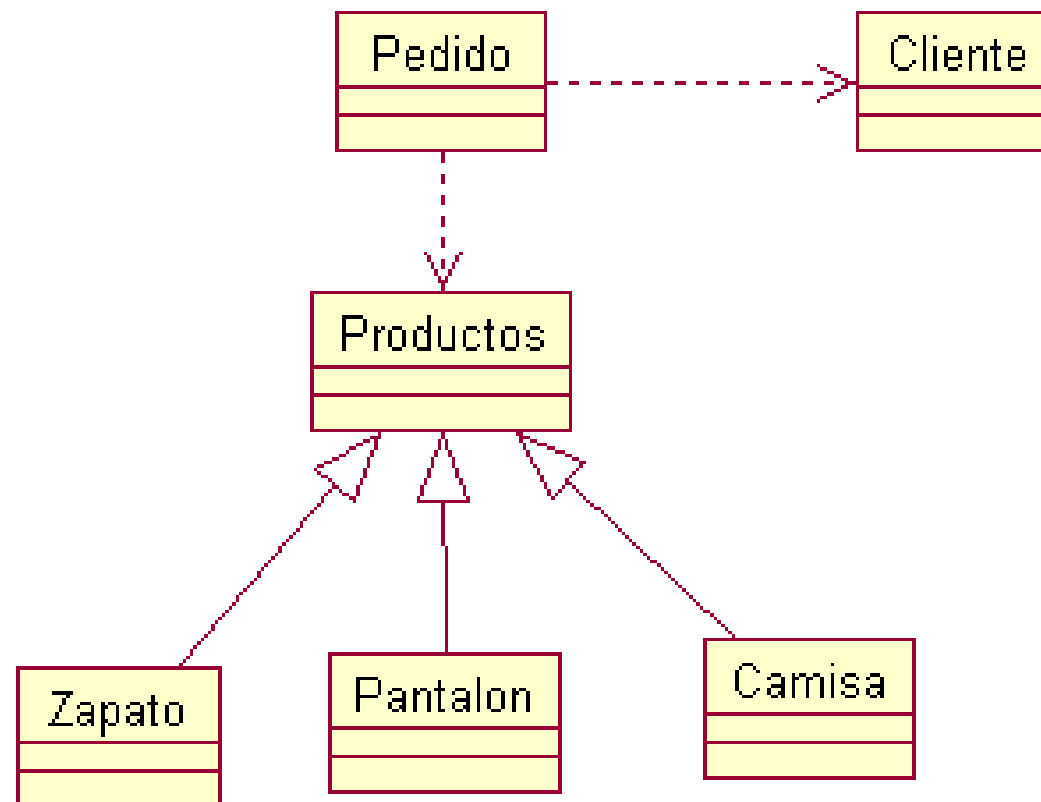
Diagrama de clases para el almacén (Alto nivel de Abstracción)



Bajo nivel de detalle

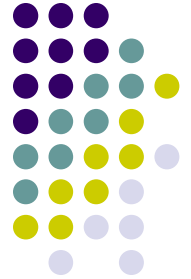


Otro ejemplo:

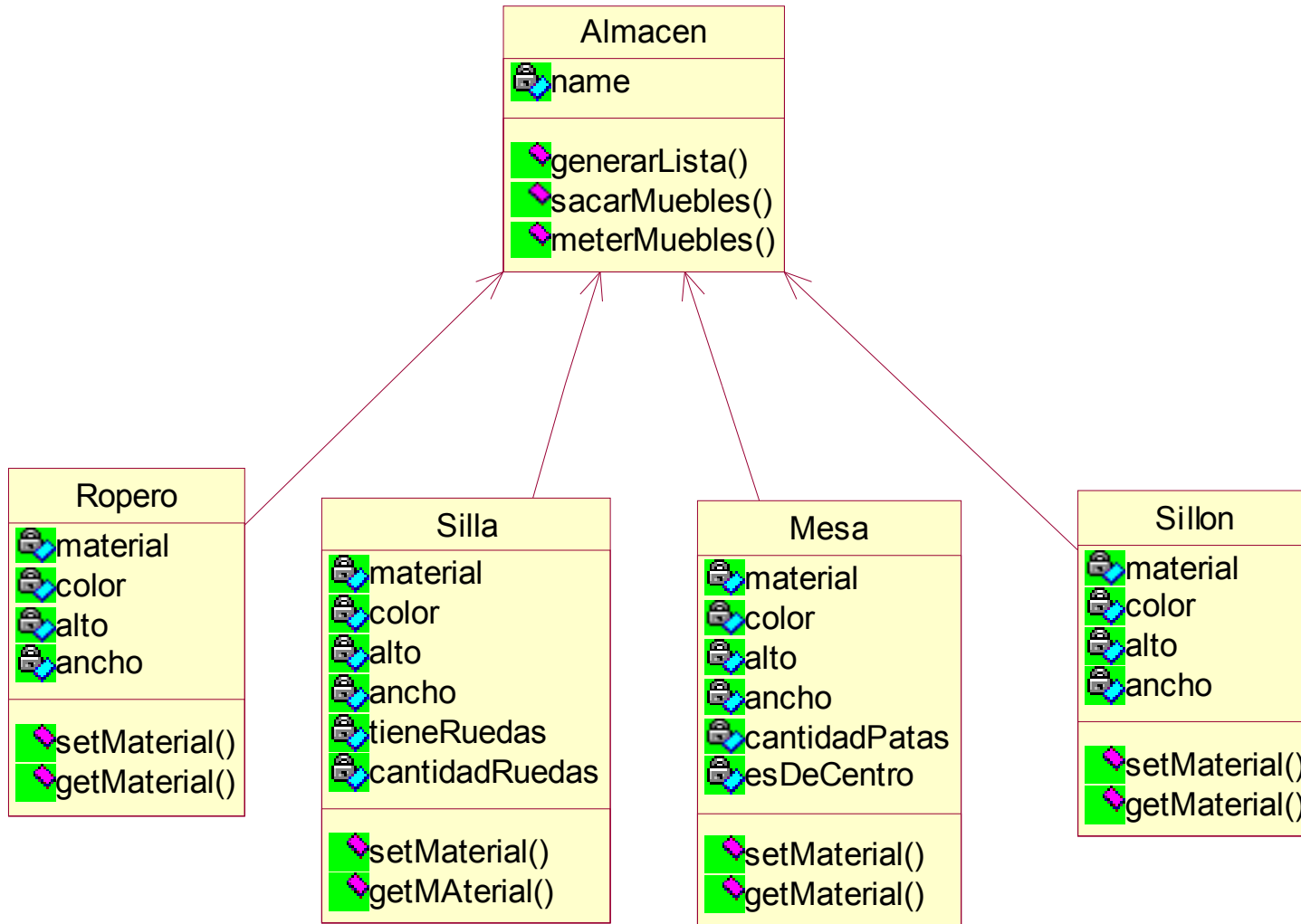
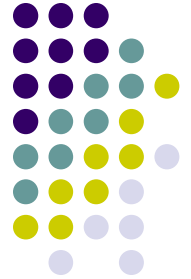




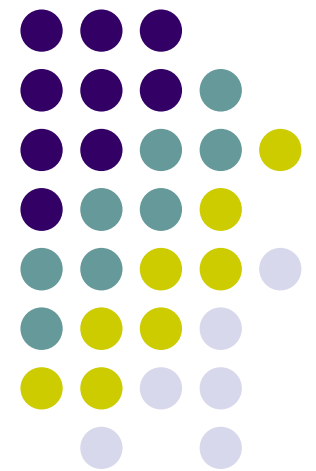
Diseño con bajo nivel de abstracción

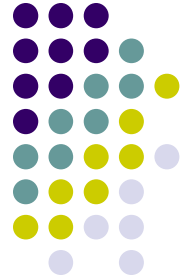


- Una vez definidas las clases y sus relaciones se procede a detallarlas.
- Se establecen sus atributos y métodos
- Para los atributos es deseable definir el tipo de dato
- Para los métodos es deseable establecer los parámetros de entrada y tipo de retorno.



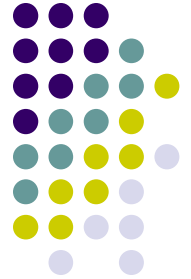
Relaciones





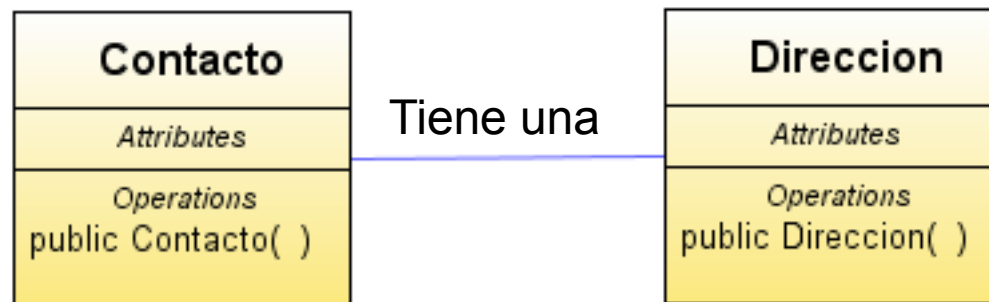
Tipos

- Como ya vimos existen relaciones entre las clases, lo tipos son:
 - Asociación
 - De Herencia (Especialización)



Asociación

- Denota la relación en cuanto a estructura de las clases que describirá la conexión que existirá entre los objetos.





Asociación: Conceptos relacionados

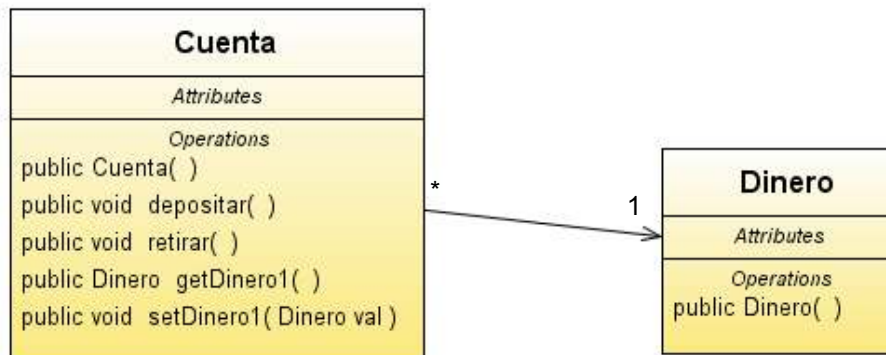


- Navegación de la asociación.
- Cardinalidad (Multiplicidad).
- Composición y Agregación.

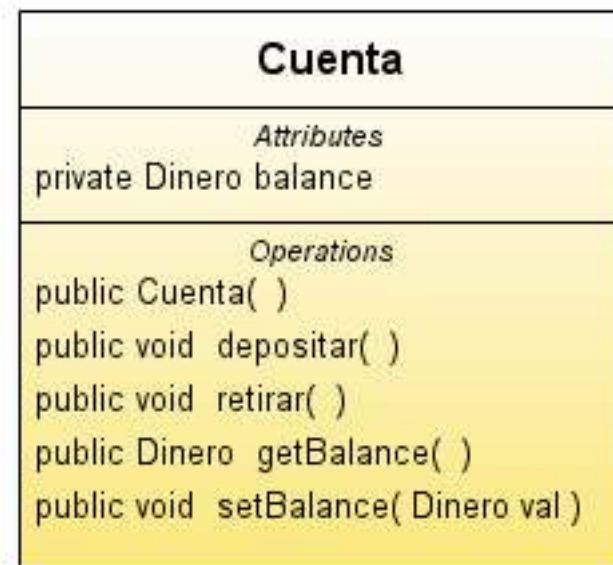
Navegación de la asociación.

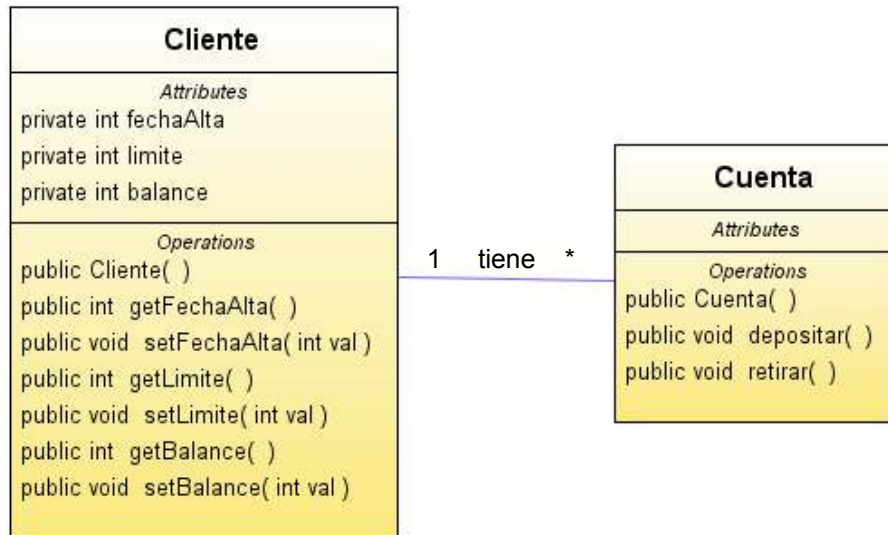
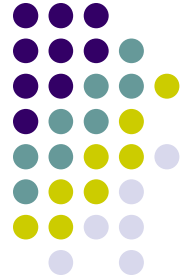


- Cuando son de línea simple se consideran bidireccionales, sin embargo para tener mayor control se recomienda establecer su navegación (dirección de la asociación).

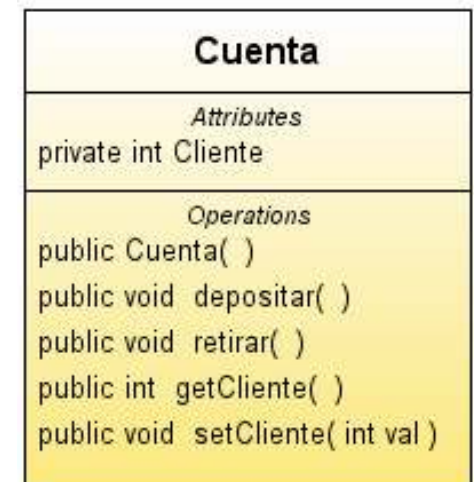
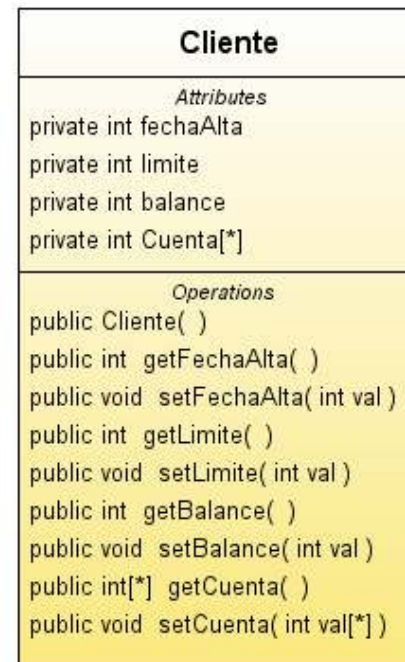


Equivale a:



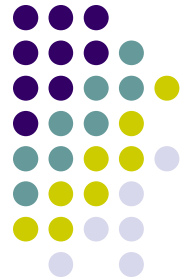


Equivale a:





Esto tiene que ver con donde se crearan los objetos

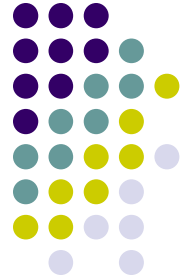


- Esto tiene que ver con donde se crearan los objetos(instancias) de la relación

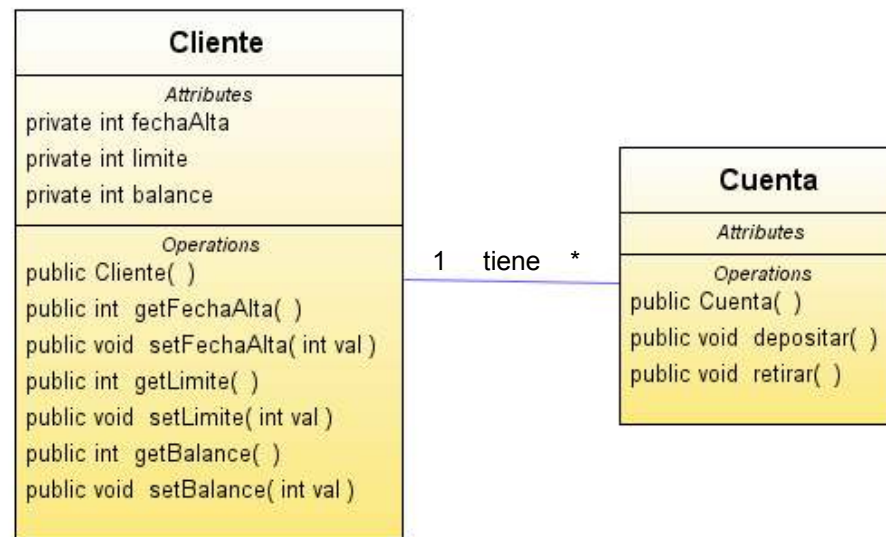
Cuenta
<i>Attributes</i> private Dinero balance
<i>Operations</i> public Cuenta() public void depositar() public void retirar() public Dinero getBalance() public void setBalance(Dinero val)

```
public class Cuenta {  
  
    private Dinero mDinero;  
  
    public Cuenta () {  
    }  
  
    public void depositar () {  
    }  
  
    public void retirar () {  
    }  
  
    public Dinero getDinero () {  
        return mDinero;  
    }  
  
    public void setDinero (Dinero val) {  
        this.mDinero = val;  
    }  
  
}
```

Cardinalidad (Multiplicidad).



- Es el número de instancias que se relacionan en una sola dirección con otra instancia

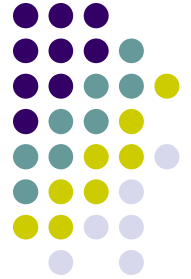


Un cliente puede tener muchas cuentas

Una cuenta solo tiene un cliente



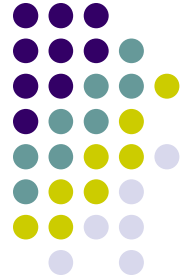
Cardinalidad (Multiplicidad).



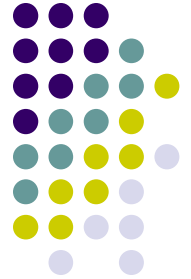
Cardinalidad	Significado
1	Solo uno
0 .. 1	Cero o Uno
*	Cero o varios
1.. *	Uno o varios (+)
n .. m	Desde n hasta m



Cardinalidad (Multiplicidad).



- Cuando la multiplicidad mínima es 0, la relación es opcional.
- Cuando la multiplicidad es mayor que 0 la relación es obligatoria



Composición y Agregación

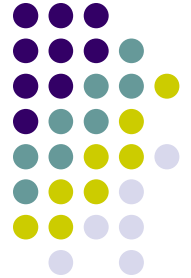
- Para comprender esto hagamos otro ejemplo:

Diseñe una clase teclado

Ahora una clase Mouse

Por último una clase Monitor

- Supongamos que estas tres clases las utilizamos para un programa X.



Monitor
<i>Attributes</i>
private String marca
private String modelo
private int pulgadas
<i>Operations</i>
public Monitor()
public String getMarca()
public void setMarca(String val)
public String getModelo()
public void setModelo(String val)
public int getPulgadas()
public void setPulgadas(int val)

Mouse
<i>Attributes</i>
private String marca
private String modelo
private String tipo
<i>Operations</i>
public Mouse()
public String getMarca()
public void setMarca(String val)
public String getModelo()
public void setModelo(String val)
public String getTipo()
public void setTipo(String val)

CPU
<i>Attributes</i>
private String marca
private String modelo
private int velocidadProcesador
<i>Operations</i>
public CPU()
public String getModelo()
public void setModelo(String val)
public String getMarca()
public void setMarca(String val)
public int getVelocidadProcesador()
public void setVelocidadProcesador(int val)

Teclado
<i>Attributes</i>
private String marca
private String modelo
private int numeroDeTeclas
private int multimedia
<i>Operations</i>
public Teclado()
public String getMarca()
public void setMarca(String val)
public String getModelo()
public void setModelo(String val)
public int getNumeroDeTeclas()
public void setNumeroDeTeclas(int val)
public int getMultimedia()
public void setMultimedia(int val)



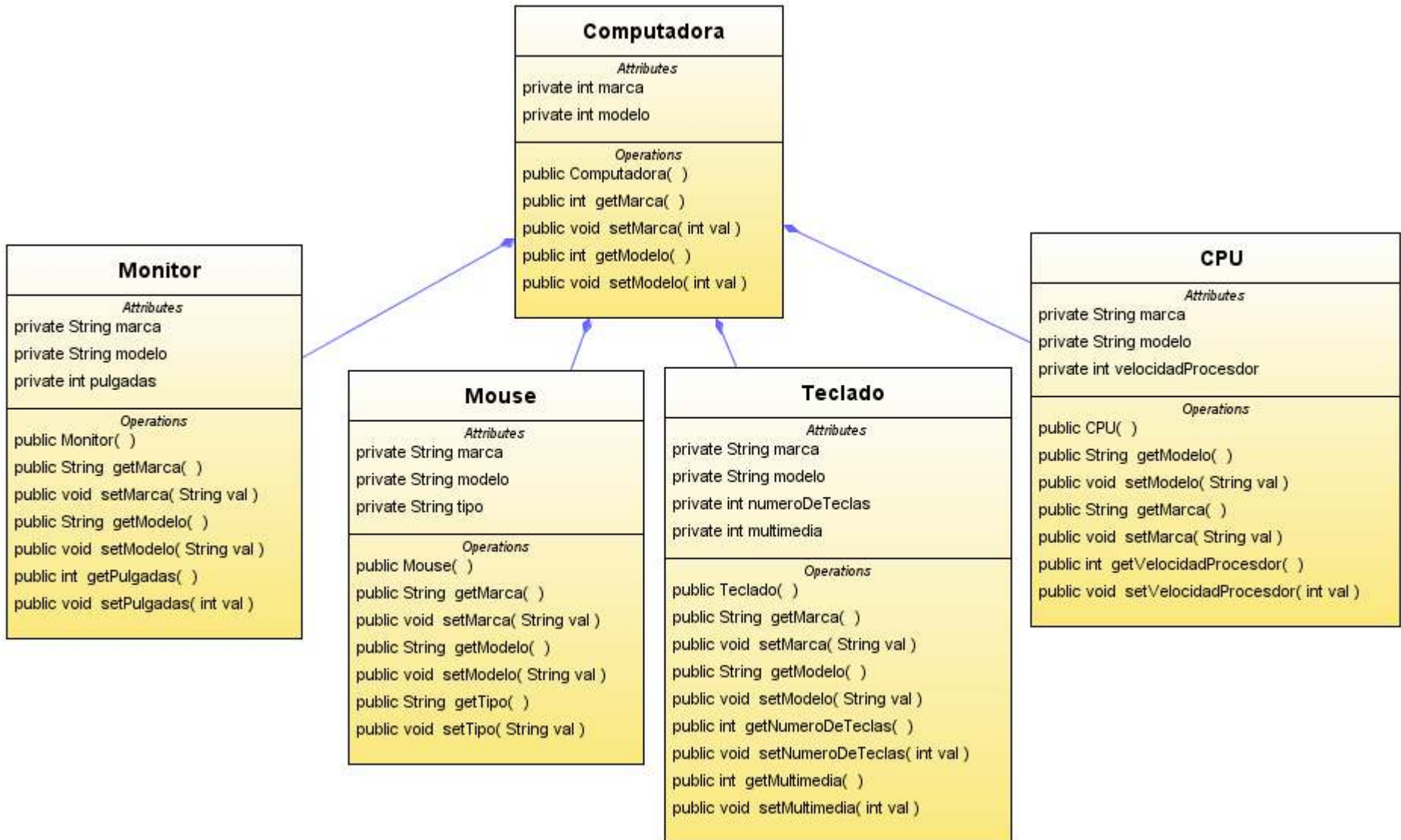
- Ahora dise

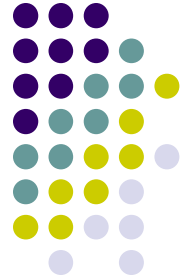
Computadora
<i>Attributes</i>
private String marca
private String modelo
private String marcaTeclado
private String modeloTeclado
private String marcaMouse
private String modeloMouse
private String marcaMonitor
private String modeloMonitor
private int pulgadasMonitor
private int velocidadProcesador
private String tipoMouse
private int numeroTeclas
private int multimedia
<i>Operations</i>
public String getMarca()
public void setMarca(String val)
public String getModelo()
public void setModelo(String val)
public String getMarcaTeclado()
public void setMarcaTeclado(String val)
public String getModeloTeclado()
public void setModeloTeclado(String val)
public String getMarcaMouse()
public void setMarcaMouse(String val)
public String getModeloMouse()
public void setModeloMouse(String val)
public String getMarcaMonitor()
public void setMarcaMonitor(String val)
public String getModeloMonitor()
public void setModeloMonitor(String val)
public int getPulgadasMonitor()
public void setPulgadasMonitor(int val)
public int getVelocidadProcesador()

nputadora



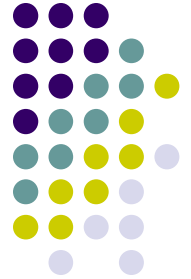
Composición





Composición y Agregación

- *Composición: Se crean clases a partir de otras, creando objetos que componen la clase nueva. **En este tipo de asociación la clase creada se encarga del ciclo de vida de los objetos que lo componen.***
- *Agregación: Es el mismo concepto, solo que en lugar de crear objetos dentro de la clase, se hace referencia hacia ellos fuera de la clase, de tal modo que el ciclo de vida de los mismos no depende de la nueva clase .*
- *Su relación es de “contiene un” .*



Herencia

- Mecanismo que nos permite reutilizar los atributos y metodos de una clase para heredarlos a una clase derivada que los especializara.
- Ejemplo las Clases Vehiculo, Vehiculo Terrestre, Automovil, Bicicleta, Avion y Helicoptero
- Su relacion es del tipo “es un”

